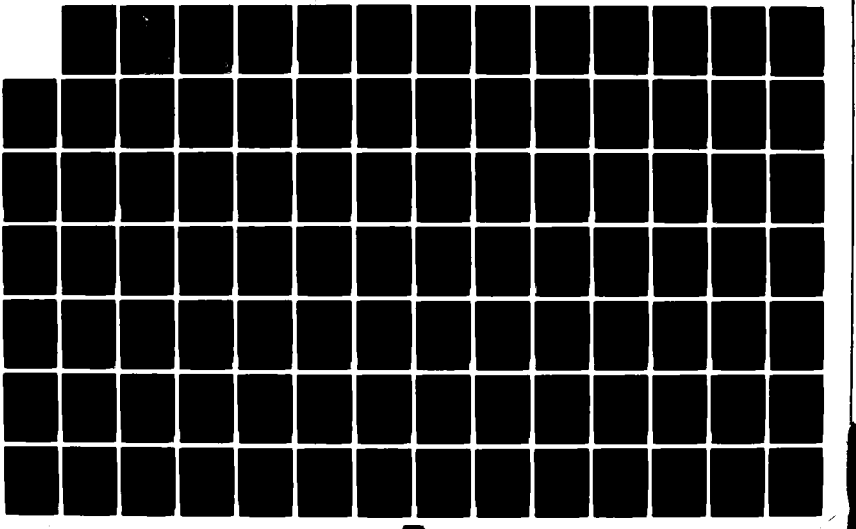
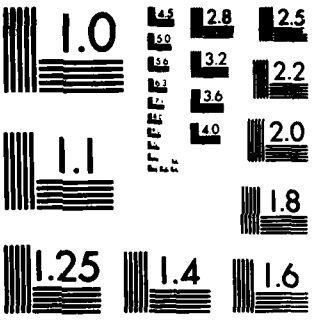
AD-A147 632 AN APPLICATION OF DISCRIMINANT ANALYSIS TO THE SELECTION OF SOFTWARE COST..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF SYST.. J T STEIG

UNCLASSIFIED SEP 84 AFIT/GSM/LSY/84S-26 F/G 14/1 NL





(2)

AD A147632

AN APPLICATION OF DISCRIMINANT ANALYSIS TO THE SELECTION OF SOFTWARE COST ESTIMATING MODELS

Jeffrey T. Steig
Captain, USAF

AFIT/GSM/LSY/84S-26

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

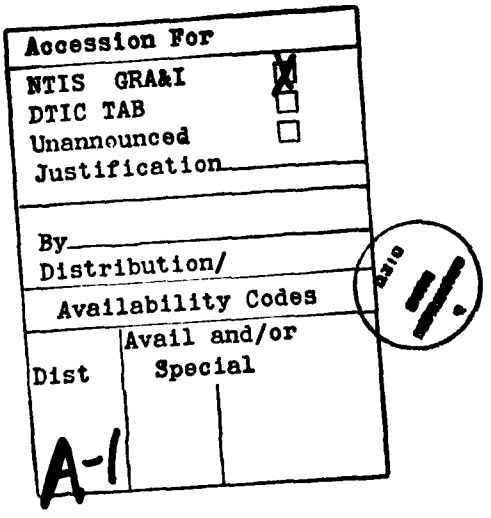
AFIT/GSM/LSY/84

AN APPLICATION OF DISCRIMINANT ANALYSIS TO THE SELECTION OF SOFTWARE COST ESTIMATING MODELS

Jeffrey T. Steig
Captain, USAF

AFIT/GSM/LSY/84S-26

The contents of the document are technically accurate, and no sensitive items, detrimental ideas, or deleterious information are contained therein. Furthermore, the views expressed in the document are those of the authors and do not necessarily reflect the views of the School of Systems and Logistics, the Air University, the United States Air Force, or the Department of Defense.



AFIT/GSM/LSY/84S-26

AN APPLICATION OF DISCRIMINANT ANALYSIS TO THE SELECTION OF SOFTWARE COST ESTIMATING MODELS

THESIS

Presented to the Faculty of the School of Systems and Logistics

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Systems Management

Jeffrey T. Steig, B.S.

Captain, USAF

September 1984

Preface

The purpose of this study was to improve the accuracy of software development costs estimations. Software expenditures are increasing exponentially while ability to estimate their costs remains only mediocre.

Although the results of the study did not indorse further use of determinants for selection of software cost models, more research area could potentially reveal a use for determinant analysis within the field of software estimating.

Several persons contributed significantly to this study. My advisor, Mr. Rich Murphy originally had the idea of applying discriminant analysis to the problem of selecting a model. Most of the data was made available by Captain Joe Dean from ESD and Andy Naiberg of TASC. Captain Art Mills, Mr. Dan Ferens and Captain Joe Dean each spent hours answering questions and lending their expertise from the field of software cost estimating. I would like to also thank the people from QSM and ECI for the unfunded use of their models. Most importantly, I deeply appreciate the untold sacrifices of my wife Shelly, who, despite the birth of our third child, assumed all my extra duties while I worked on AFIT graduation requirements.

Jeffrey T. Steig

Table of Contents

List of Figures

List of Tables

AFIT/GSM/LSY/84S-26

Abstract

This investigation attempted to improve software cost estimating through the use of discriminant analysis. Currently, no quantitative methods exist to quantitatively select the best software cost estimating model for a particular software type or environment. By identifying the characteristics of the software that each model was best able to estimate, those characteristics could be used as a basis for predicting the best model.

The analysis began by using selected models to concurrently estimate development costs for 25 known projects. The estimates from each model were compared and the most accurate model for each project was identified. The projects were assigned to the group of projects for which each model most accurately estimated development costs. The groups were given the names of the respective models: PRICE S, SLIM, COCOMO, and JENSEN.

After grouping each project, discriminant analysis was used to identify those input variables from all the estimating models that best discriminated between the groups. The identified input variables were then used as determinant variables as a basis to predict which model was most likely to best estimate cost for each project. The unbiased prediction rate was 60%. Despite the high prediction

rate, the overall estimating accuracy was not reduced. The criteria statistic, relative root mean squared error, RRMSE, for the estimates selected by discriminant analysis was not significantly better than the RRMSE for the expected error. The results indicated that the use of the pre-analysis determinants to select a model would not reduce estimating error more than a random selection of models.

AN APPLICATION OF DISCRIMINANT ANALYSIS TO THE SELECTION OF SOFTWARE COST ESTIMATING MODELS

I. Introduction

Justification

Skyrocketing software costs show no indication of subsiding. In 1980, 40 billion dollars were spent on software in the United States (about 2% of the GNP). Projected growth for 1985 and 1990 is to 8.5% of the GNP and 13% of the GNP respectively (1:3). Unfortunately, software cost estimating capabilities did not develop adequately as the industry expanded. Current cost models rate well if they predict within 20% of the actual cost in 70% of their cases (3:32). Two reasons can be offered to explain the difficulty in estimating software costs. The first reason, a temporary one, will eventually pass with time. Software development engineering is a relatively new discipline and historical data bases for cost estimating are almost nonexistent (36:1-5). The second reason for estimating difficulty in software lies in the nature of the commodity itself, and it will not diminish with time. Unlike most products for sale, software is not a material good. Instead, software represents the ideas from the mind of a computer programmer. Nebulous products, such as ideas, resist specific quantification, measurement and cost prediction. The result

of this resistance to quantification and measurement has had detrimental effects on Air Force research and development efforts.

> In addition to the constantly rising cost for software development, software reliability, unresponsiveness, and indirect costs associated with slippages in software developments are of major concern to the USAF. (16:1)

Problem Statement

Numerous software cost estimating models are currently available and each has experienced some degree of success in estimating costs for software (1:3). There is not, however, a tool to help cost analysts select the best cost estimating model for their particular software project. Generally, cost analysts currently depend on one cost model to predict all software costs. In the cases when more than one model is available, no method exists to indicate which model's prediction may be more accurate. This forces the analyst to heuristically decide whether to develop a composite forecast incorporating all the outputs from the available models or to favor one estimate because of the models' past performance. A good example of the latter exists at Electronic System Division (ESD). Captain Joe Dean, the chief software cost analyst at ESD, tends to favor the results of one particular model when estimating the software development costs for embedded avionics (6). He bases his choice on favorable past

experiences with that model. An empirical method for selecting the best cost model could improve the cumulative accuracy of all the estimates. This supposition follows naturally from the conclusions of several comparative studies (7; 8; 22; 36; 37). Each of these studies concluded that both the environment in which the software is designed to operate and the environment in which the cost model was developed significantly affect the relative estimating performances of the models tested. Randall W. Jensen made a similar comment in a recent article on software cost estimating.

> There are few model comparisons available which allow the user to judiciously select the estimation model best suited to a specific project type or organizational approach. (19:97)

A cost model developed from embedded avionics software data would tend to return more consistent cost predictions for embedded avionics applications. Additionally, the studies also indicated that across a range of software environments, the accuracy for each cost model was inconsistent, even for the most accurate cost models. In one case, the model with the most accurate cost prediction in two software environment categories was the worst estimator in the third environment (36:5-27). None of the comparison studies established the consistency of the cost models' accuracy as a function of software environment. But potentially, the ability to select the most accurate cost model could eliminate the outlier cost predictions and

significantly improve overall software estimating results. The analyst could then select the best model based on software program characteristics and would not be forced to average estimates or rely on intuition. This technique would combine the strengths and minimize the weaknesses of the current state-of-the-art software cost estimating models.

Scope of Research Effort

This research effort will limit its investigation to cost models which would be the best candidates for estimating development costs at Aeronautical System Division (ASD). Development schedule and support costs will not be addressed. Two criteria were used to select the models. First, the model must already be in use within the Air Force for software cost analysis. And second, the model must have a good record of accuracy with its Air Force users. The four models which best fit these criteria are: RCA PRICE S, a proprietary software model based in New Jersey (29); Constructive Cost Model (COCOMO), a non-proprietary software model developed by Dr. Barry Boehm (3); Putnam SLIM (SLIM), a proprietary software model developed by Lawrence Putnam (28); and the Jensen Model, developed by Randall Jensen (18).

Three questions will be addressed in the research.

1. Do software development cost estimating models accuracy correlate to any specific software characteristics?
2. Which determinants best differentiate software

characteristics and can they be used by analysts to select the cost model which will return the most accurate estimate?

3. Will this ability to predict the most accurate cost model significantly improve software cost estimating?

Hypothesis

The need exists for a quantitative method of selecting the estimating model best suited for a particular situation. Through the use of pre-analysis determinants it will be possible to determine the software characteristics that correlate to the estimation accuracy of the tested models. Furthermore, this technique will reduce the total estimating error by selecting the estimating model most likely to provide the most accurate cost estimate.

II. Literature Review

Three topics will be covered in the literature review. First, a brief review will be made of the literature which substantiates the current software cost estimation problem. Second, those studies which have compared software cost estimating models will be reviewed. Finally, justification will be made for the models selected for this thesis.

Problem Justification

The importance of software cost estimating has increased as software costs have increased (14:1018). The Department of Defense estimates that by 1990, 85% of the budget spent on computer related acquisitions will be spent on software (23:5-S). However, within this area of increasing importance rampant cost overruns are not unusual. In his thesis, Capt Devenny related how the U.S. Air Force command-and-control software project had four changes in its estimate of cost. The final cost was ten times the original contractor's "best estimate" (7:19). Accurate cost estimates can and should be the critical element for software development efforts by enabling acquisition managers to properly allocate time and resources for software (33:1). Numerous other examples of cost overruns, schedule slippages and cancelled programs have been associated with poor software cost predictions. Cost model developers have concluded that despite the progress

made to date, a need for more accurate cost estimating results continues to exist within the software industry (34:18; 14:1018; 36:7-2). The state-of-the-art of software cost estimating currently is only mediocre. The best software cost estimating models are only within 20% of the actual, and they do that in only 70% of their predictions (3:32). Why are software costs difficult to consistently predict?

A common reason identified in the literature for software cost inconsistency is the diversity inherent to software (33:2). This diversity comes from two sources. First, software can be remarkably different in its application and internal structure. For example, the type of software needed to perform accounting calculations varies greatly from the software needed to control the flight of the Space Shuttle. The second source of software diversity is the environment in which the software can be developed. A few examples of software environment are software engineering methods, supporting software, type of computer used and type of contract (36:5-27). Because of its two-fold diversity software presents a difficult commodity to estimate. Isolating the most critical cost factors can be a difficult task. Ninety-six cost drivers for software were enumerated in one study (25), and 53 in another (15). With the potential variations so immense, Bailey and Basili questioned if one model could hope to cover all the possible combinations (2:107). Thibodeau essentially indicated the same idea when

he concluded that no one software cost estimating model met U.S. Air Force requirements for accuracy. Thibodeau stated further that "this finding indicates the necessity of learning the specific attributes of a development environment that determine when one or another model structure should be used." (36:5-27)

Rather than follow Thibodeau's advice, many software cost estimating model developers tried another tactic to improve estimations. Several developers combined two or more existing models in an attempt to manage the diversity in software. (2; 4; 8; 21; 31; 35). The logic of combining cost models is appealing. Theoretically, the product of this process would have the best features of each of the individual models. The literature does not, however, include any empirical data to indicate the combined models have improved cost estimating abilities. In two attempts to combine models more complexity resulted rather than improved handling of diversity. The combined estimating models became too complex for routine cost estimations and consequently their use was discontinued (4; 5; 31).

This thesis will not attempt to combine models; instead, it will follow the direction indicated by Thibodeau. This effort will seek "to determine when one or another model should be used". The research will then be taken one step further. Pre-analysis determinants will be used to predict the model which will make the most accurate cost estimate for the software under consideration. These determinants should

allow the analyst to improve model selection, and reduce the total cost estimating error. Before beginning the methodology, it will be instructive to first review those studies in the literature which have compared software cost estimating models.

Review of Model Comparison Studies

Literature on software cost estimating models per se can be found in abundance. This proliferation of analysis and comment stems from the substantial concern the topic warrants, especially within the defense industry. Most studies on the topic of software cost estimating spend at least some time reviewing other cost estimating models. Because ample descriptions and illustrated uses of popular cost models abound in the literature, this review will not needlessly repeat what has been done elsewhere. Instead, this review will concentrate on studies that compare software cost estimating models. Comparison studies can be classified into two categories: qualitative word studies and quantitative empirical studies.

Qualitative Comparisons. The qualitative word comparisons of software cost estimating models primarily sought to inform the reader about the characteristics of the cost models available. General overviews of available cost models can prove to be quite helpful to orient readers concerning the current state-of-the-art in software cost

estimating. Boehm (3) and James (15) cover most of the popular models and provide a cursory look at each model and the underlying methodology used to develop each. Boehm comments expertly on the strengths and weaknesses of each model reflecting his extensive background within the software development industry. He did not comment on the Jensen model used in this study because it had not been published at the time Boehm's work was printed. Included in Boehm's book (3) and Dirck's article (8) are two helpful tables concisely diagraming the input parameters for each of the models covered in their text. The tables list the range of input factors that the cost estimating models use to estimate the cost of software. In addition to listing input factors, several studies delved deeper and included evaluations of both input factors and output factors (7; 24; 30). These studies critically compared the usability of input factors and the utility of output factors with respect to user requirements. Both informative and thought provoking, the qualitative word studies adequately provided conceptual comparisons of the cost models. But a prospective cost model user can also use empirical data to compare the accuracy of existing models. This type of research was done in the quantitative comparison studies.

Quantitative Comparisons: McCallam. Two articles reported empirical results from comparisons of software cost estimating models: McCallam and Thibodeau (22; 36). McCallam compared three models' ability to estimate software costs by

using them to predict cost for a single software program. After describing the software program that each model estimated, McCallam detailed how each model was calibrated and used to estimate the cost. He specifically selected the three models because of their contrasting methodologies and input variables. The first model was a top down software sizing model which the author co-developed with another researcher. The second and third models are well know proprietary models (SLIM and RCA PRICE S). After explaining each of the input parameters, McCallam calculated the bottom line for cost and expressed it in manmonths, a commonly used expression of software cost which readily permits comparison of model accuracy. The results from each model were compared to an inhouse estimate from the developers of the software program. The results proved interesting: all three models predicted costs within three percent of the in-house estimate. Those are surprisingly accurate results. McCallam pointed out the advantage of using more than one model to predict costs for software and stated further that the more diverse output of information from the three models could aid in managing the software development. However, an obvious problem was not addressed. What should the analyst do if the three models do not estimate the costs to be approximately equal? It would be left up to the analyst to determine which estimate to favor. Because only one program was estimated, the results from this study cannot be relied on as indicative for other types of software. The second empirical study

sought to provide more transferable results by using several types of software in its comparison.

Quantitative Comparisons: Thibodeau. Thibodeau conducted a detailed empirical comparison of nine software cost estimating models evaluating their potential for use by the United States Air Force (36). That research effort constitutes the most extensive and rigorous comparison of software estimating models in the literature. Two portions of that study will be reviewed: model definition and evaluation.

The first portion of the Thibodeau report, model definition, resembles the type of qualitative word comparisons conducted by other studies. The model definitions consisted of three parts: description, definition of input/output variables, and classification. In describing the models Thibodeau repeated in his study the most common complaint of the other software cost estimating studies. That is, with only a few exceptions, cost estimating models lack sufficient documentation to adequately explain what specific aspects of software development efforts are considered and predicted by their models. In some cases the researchers were required to contact the software model developers in order to fully understand and explain how and what the cost estimating models specifically included in their predictions of software costs. This background was essential to ensure that the models were compatible with the government's requirements for sufficient comprehensiveness.

For example, considerable discussion was devoted to the model output compatibility with both the phases of defense acquisition (conception, validation, and full scale development) and levels of the Work Breakdown Structure. In addition to the description of the models, Thibodeau classified each into categories according to their underlying method of evaluation. The three classifications of software cost estimating models were regression, heuristic, and phenomenological. Predictably, the regression models derive their estimates from linear and non-linear regression techniques; the heuristic models utilize both regression, interpretation and supposition; and the phenomenological model (only the SLIM Model fit this classification) incorporates a concept which is not limited to describing the mechanics of software development (36:4-16). The phenomenological concept, learning rate, assumes the rate which programmers learns is constant. After thoroughly defining the models, Thibodeau began the most unique part of his study, an empirical evaluation and comparison of the models.

The evaluation had two important portions: use of data and findings. In the first portion, Thibodeau used data representing three different types of software. The types of software represented were commercial data management, groundbased satellite support, and United States Air Force data management. The second important portion of the evaluation was the findings. Although nothing came out of

the Thibodeau study that was not already in the literature, the findings are important because they empirically verified the suppositions posed by other software cost model researchers. The highlights of the findings follow. 1) *Calibration of the model to the data set substantially* affects the accuracy of the results. 2) The development environment significantly affects the relative performance of the cost estimating models. 3) Lack of good data on software development limits the ability to create and evaluate software estimating models. 4) No models currently available adequately predict cost for software. The first and fourth of these findings point toward the work proposed in this thesis. Software cost estimating needs to be improved; and the environment significantly affects model performance. Developing pre-analysis determinants to select the best model should strengthen both of these weaknesses.

Model Selection

The models used in this effort were chosen because they were the best candidates for use in ASD. Two criteria were used to evaluate the available models. First, the model should currently be used within the U.S. Air Force for software acquisition. Second, the model should have a strong record for software cost estimating.

The first criteria, requiring the software cost estimating model to be used within the Air Force, ensures

that all models used will be compatible with the defense acquisition process. It is not a serious limitation that this criteria may eliminate some currently unknown models. The purpose of this effort is not to exhaustively present the options available for software cost estimations; instead, the purpose is to indicate a technique which will improve the use of the models currently available.

The second criteria, requiring that the models have a strong record, is self-explanatory. More can be gained by combining the strengths of the more effective models than can be gained from combining the strengths of the less effective ones.

Four models fit within these criteria: RCA PRICE S (RCA) (29), Constructive Cost Model (COCOMO) (3), SLIM (28), and Jensen (19). Interviews with a cost analyst from each of the four product divisions within the Air Force System Command, (Aeronautical System Division, Electronic System Division, Space Division and Armament Division) established the selected models as those most often used and those most accurate in past performances (5; 11; 20; 32). Major Joe Duquette, the cost analyst on the Air Staff responsible for software cost estimating, also confirmed that the four models selected for this project were the best candidates for a comparison study.

III. Methodology

The methodology followed a three step process. The first step used all four software cost estimating models to estimate costs on twenty-five known software projects. Each project was assigned to a group, depending on which model most accurately estimated its development cost. In the second step, discriminant analysis was used to identify characteristics of the software projects which correlate to the models and the projects they most accurately estimated. The final step evaluated the success of pre-analysis determinants to reduce estimating error in software cost estimating.

Cost Estimating Procedure

In the cost estimating procedure, input variables from the four selected models were derived from information on the twenty-five know software projects. Descriptions of the four estimating models along with the input data can be found in Appendices B and C respectively. In addition to the background information found in the appendices two areas need clarification regarding the testing procedure. The first area involves the methods used to standardize the models' inputs and outputs to enable comparisons. The second area pertains to the test data sets used in the research. The types and sources of the test data sets are described in the second section.

Data Set and Estimating Model Definitions. Lack of quality data plagues software cost model development (12:175). Data often fails to consistently reflect the assumed attribute. Size and development are two such parameters whose meaning can vary.

Measurements of program size vary considerably. Common examples of these include: lines of delivered source instructions (3:58), number of source statements, and lines of object code (36:4-5). Confusion can arise if the size measurement variable is not correctly identified for the particular model being used. In this research, a conversion factor was used for projects 13-25 to convert lines of source code into lines of object code for use in PRICE S. The translation equation from the PRICE S Users Manual (29:III-25) was used with a slight modification. Based on the experience and advice of Captain Joe Dean, Electronic System Division (ESD) Software Cost Analyst, the expansion factor used in this research was EF = [(EXP-1) / 2] + 1, where EF is the expansion factor used in this study, and EXP is the expansion factor listed in the PRICE S User's Manual.

Another important factor needing explanation is development cost. Comparing the cost in dollars causes problems because of the time value of money, inflation rate and other monetary problems. For that reason, the comparison variable used in this research will be the amount of development effort required, not dollar cost. The effort

measurement will be in man-months (152 man-hours). PRICE S's outputs in dollars were translated into man-months for projects 13-25 using the conversion of one man-month equal to $8,750.00. This conversion factor was recommended by Mr. Earl King, RCA PRICE Customer Relations.

Test Data Set. The data used in this study came from two sources. The first twelve projects came from the Thibodeau report (36). Projects 1 through 6 were ground-based satellite support projects. Projects 7 and 8 were Air Force Data Systems Design Center projects developed for payroll and logistical applications. Projects 9 through 12 were commercial data base management system projects. The Thibodeau study only provided input variables and estimates for PRICE S and SLIM. Input variables on the first twelve projects for COCOMO and the JENSEN model were derived from the inputs listed in the Thibodeau report for SLIM and PRICE S.

The second source of data, projects 13-25, came from an unpublished report by The Analytical Science Corporation (TASC). On contract with ESD, TASC collected data on twenty-eight projects using the methodology from a recent data collection study done by the Mitre Corporation (8), called Software Acquisition Resource Expenditure (SARE). The study recommends a format and identifies the factors needed to effectively capture critical software cost drivers. The SARE methodology provides uniform data gathering and of parallel input variables needed for cost model comparisons.

Data on some projects collected by TASC contained omissions on development schedule and other key variables. These omissions prevented the use of fifteen of the projects in this study. Additionally, because of the missing schedule information there was no way to calibrate the models uniformly. To standardize the model estimates each model was required to internally calculate any variable normally provided through a calibration process. The only exception to this rule was the variable RESO for PRICE S. On the advice of Mr. Earl King of RCA, RESO was set at 3.3, the defense industry average. Where minor omissions occurred in the thirteen projects used from the TASC report, the default or nominal value was used to minimize to adverse effects on the estimate.

After the four cost models were used to estimate costs for the twenty-five known software data inputs, the results were used in the second step of the methodology, discriminant analysis.

Discriminant Analysis

The discussion of discriminant analysis will include four topics. Following a brief description of discriminant analysis, its inputs and outputs will be discussed. Finally, the criteria used to select the pre-analysis determinants will be offered.

Description of Discriminant Analysis. Because discriminant analysis is not often linked to comparisons of software cost estimating models, a brief description of its function in this research will be given. The discriminant analysis programs used in this research were from the Statistical Package for the Social Sciences (SPSS)(26) and the Bio-Medical Decision Package (BMDP)(9). A more complete description of the statistical background of discriminant analysis can be found in Appendix A.

Functionally, discriminant analysis attempts "to statistically distinguish between two or more groups of cases" (26:435). " To distinguish between the groups the researcher selects a collection of discriminating variables that measure characteristices on which the groups are expected to differ" (26:435). All of these discriminating variables can be entered as potential discriminators. The program statistically evaluates each of the variables according to their ability to discriminate between the groups (26:435). In this study, the statistic called Mahalonobis D will be used to evaluate each variables discriminating ability. The Mahalonobis D is reflected by the F-to-enter value (26:447). After selecting the variables to be included in the discriminant function it can be determined what percent of the software projects could have been correctly classified (26:436). The variables included in the equation will be the pre-analysis determinants. Based on the values of the variables, a prediction can be made as to which group

a particular software project belongs (26:445).

Discriminant Analysis Inputs. The inputs for the discriminant analysis will include both the inputs and the outputs of the first research step. The twenty-five software projects evaluated by the selected cost models will be categorized into four groups: COCOMO, PRICE S, JENSEN and SLIM. Each group will contain the software projects for which each respective cost model most accurately estimated development cost. The discriminant variables used for the analysis, will include the input variables used by all the cost models to derive the original estimates. This procedure will ensure that the same information already gathered to perform cost estimates can also be used to predict the most accurate cost model. Logically, the most relevant cost drivers would be among the factors used to estimate costs. These same factors should be the best discriminators to predict the most accurate software cost model for each software project. With this input, the discriminant analysis program will provide the necessary output for predicting the most accurate model.

Discriminant Analysis Outputs. The output from the discriminant analysis will include: 1) the discriminating variables most effective in predicting the most accurate cost model for each software project; 2) a discriminant function which, based on the discriminant variables, will maximize the ability to discriminate between the groups; 3) a classification function which will classify each case into a

model group based on the values of the discriminant variables and 4) the percent of cases correctly classified. All four outputs will be used in an iterative process to select the final variables used as pre-analysis determinants in the final set of discriminant functions.

Selection Criteria for the Discriminant Variables. The combination of discriminant variables selected will the ones whose classification function is most accurate and stable. The accuracy will be measured by the percent of the known projects correctly classified by the classification function. The stability will be measured by the difference between the jackknifed and unjackknifed classification score. The jackknifed and unjackknifed classification methods require further explanation.

The Unjackknifed method of classification produces a biased classification score. The biased or unjackknifed method, calculates the discriminant function using all twenty-five projects. The bias occurs because the discriminating function contains information about the specific case it is attempting to classify. The resulting unjackknifed score is overly optimistic. The second method of classification reduces this problem (9:711).

The unbiased classification method jackknifes the data to reduce the bias in the classifying process. In the jackknifed method, the project being classified is removed from the data set and the remaining 24 projects are used to calculate the discriminant and classification functions (9:731). The project

in question is then classified with a function without information from the case being classified. The jackknifed method produces a different set of discriminant and classification functions for each project classified. This produces a problem when the jackknifed and unjackknifed classfication scores differ considerably.

When the unjackknifed score is considerably higher than the jackknifed score, the discriminant function is considered to be unstable. This instability usually results from the discriminant function chasing random error in its attempt to differentiate between the groups. In those cases, the discriminant function changes erratically as individual cases are removed from the data. Only one set of discrimininant functions can be used for predictive analysis. Because the jackknifed method produces many discriminant and classification functions, the unjackknifed function is used as an approximation of the twenty-five jackknifed functions. The assumption of similarity between the jackknifed and unjackknifed functions is lost when the discriminant functions appear unstable.

The combination of discriminant variables will be selected that demonstrates the greatest accuracy and stability. It will be the combination that has the highest jackknifed classification score, and that appears to be the most stable.

Success Criteria

Two measures were used to report the success of the pre-analysis determinant technique. The first measure is the number of times the most accurate software cost estimating model was chosen for each of the software projects. This measure is reported in a straight percentage.

The second measure more functionally measures the success of the selection technique. The total performance for each model will be calculated by using the Relative Root Mean Square Error (RRMSE). The RRMSE was used in the Thibodeau (36) empirical comparison because it penalizes for extremely poor estimates. The measure is computed as follows:

$$RRMSE = \frac{[1/N \Sigma (ACT_i - EST_i)^2]^{1/2}}{1/N(ACT_i)} \qquad (1)$$

Where

ACT_i = The measured size of the ith project in the sample set;

EST_i = The estimated size of the ith project;

N = The number of projects in the sample.

The RRMSE is also called the coefficient of variation ($S_{y/\bar{y}}$) and measures the relative variability of the distribution around the mean. It was calculated for all of the estimates which the pre-analysis discrimination predicted to be most accurate. To compare the discriminated

performance to a random selection process the mean absolute deviation (MAD) was calculated for each project estimate. The MAD was calculated for each project by summing |ACT-EST| for the four estimates of each project and dividing by four. The RRMSE(Random) expresses the RRMSE an analyst could expect if estimates from the four models were selected at random. Comparing the RRMSE(Discriminated) to the RRMSE(Random) reveals the improvement in estimating error the pre-analysis determinants produced. A further comparison of the RRMSE (Discriminated) with each of the models' RRMSE indicated the potential for discrimination to improve accuracy over any one estimating model.

IV. Findings And Analysis

The search for determinant variables proved fruitful. Four software model input variables were statistically capable of discriminating between the members of the model groups. However, despite the ability to select the most accurate model in a majority of the cases, the technique of using pre-analysis determinants did not improve overall estimating accuracy. The presentation and analysis of the results will begin with the steps used to select the determinants. An analysis of the resulting discriminant function will follow along with an interpretation of the classification functions.

Selection of the Determinants

Selection of the determinants involved the discriminant analysis programs from both the Statistical Package for Social Sciences (SPSS)(26:434-467) and the Bio-Medical Decision Package (BMDP)(9:519-537). This discussion will assume a basic understanding of discriminant analysis. More background information on discriminant analysis techniques is in Appendix A.

The stepwise method in BMDP was used to select the most discriminanting variables. The stepwise process begins with no variables in the discriminant function and tests each variable to determine which one has the most discriminating

power. The discriminating power, or the statistical ability of the variable to explain the variation between the groups, was expressed by the F-to-enter calculated for each variable. The larger the F value, the greater the discriminating power of the variable. On the first step, the analysis selected the variable with the largest F. On the second step, a new F-to-enter was calculated based on the ability of the remaining variables to explain the variation between the groups, given that the first variable was in the equation.

In the process of selecting variables with the highest F-to-enter value, three variables with high F values were initially excluded from selection: RESO, HOST, and TURN. RESO comes from PRICE S and stands for resource. HOST comes from the Jensen Model and represents the difficulty in converting from the host development computer to the target computer. TURN comes from COCOMO, and reflects the time from computer programmer initial input to hardcopy turnaround. These variables were statistically significant initially. However, their actual values were predominantly unknown, and default or nominal values had been substituted in their place. Attempting to characterize a software project for group discrimination with a default value would be meaningless. While statistically significant for discrimination, these three variables contained no information about the software project's characteristics. RESO, HOST and TURN eventually became insignificant as other variables were entered into the

function. Apparently, the variation explained by RESO, HOST and TURN could also be explained by other variables.

Table I summarizes the variables as they were included. As indicated, various combinations of the more discriminating variables were tried to discover the most effective combination.

The variables chosen as determinants were CPLX, TECHFTR, UTIL and RELY. CPLX and RELY come from COCOMO, TECHFTR comes from SLIM and UTIL comes from PRICE S. This particular combination produced the best classification results with the fewest variables. When PLTFM was included, the classification score remained unimproved. Although ACAP improved the unjackKifed score, because the jackKnifed score decreased, the discriminant function was much less unstable.

All of the final four selected variables CPLX, TECHFTR, UTIL and RELY reflect some component of complexity or increasing difficulty with respect to software development. CPLX and TECHFTR measure the application complexity for their respective models. UTIL indicates the percent that the software project utilizes of the memory and or time constraints of the target computer. As UTIL increases from .6 to 1.0, the effort required to develop the software increases exponentially.

TABLE I

Summary of Stepwise Classification Scores

Included Variables	Correct Classification Scores Unjackknifed	Jackknifed
CPLX	20%	20%
CPLX,TECHFTR	56%	44%
CPLX,TECHFTR,PLTFM	56%	44%
CPLX,TECHFTR,RELY	60%	44%
CPLX,TECHFTR,ACAP,PLTFM	64%	52%
CPLX,TECHFTR,RELY,UTIL	64%	60%
CPLX,TECHFTR,RELY,UTIL, PLTFM	64%	60%
CPLX,TECHFTR,RELY,UTIL, ACAP	68%	52%

Variable	Model	* Variable Description
CPLX	COCOMO	complexity of the software
TECHFTR	SLIM	technical factor
UTIL	PRICE	utilitization of computer memory
RELY	COCOMO	reliablity of the software
PLTFM	PRICE	platform
ACAP	JENSEN	analyst's capability

* Appendix B has a more detailed description

RELY reflects the degree of reliability required of the software. The highest value of RELY is assigned to software whose failure could jeopardize human life. Lower values of RELY reflect software whose failure will only be a minor inconvenience. The four determinants measured similar characteristics of the software projects.

Interpretation of the Discriminant Function

A pattern was discovered in the mean values for the variables within each group as seen in Table II.

TABLE II

Mean Values for Variables by Model-Group

	UTIL	CPLX	RELY	TECHFTR
PRICE	.650	1.13	1.05	9.17
COCOMO	.667	1.25	1.32	10.67
SLIM	.700	1.21	1.20	8.00
JENSEN	.600	1.08	1.05	8.25

For each of the variables, except TECHFTR, the higher values indicate more complex, or effort intensive software characteristics. TECHFTR has lower values for more complex software. Each group shows a general trend in the relative levels of complexity. Table III illustrates this point.

TABLE III

Relative Levels of Complexity by Model-Group

	UTIL	CPLX	RELY	TECHFTR
Most complex, effort intensive	SLIM	COCOMO	COCOMO	SLIM
	COCOMO	SLIM	SLIM	JENSEN
	PRICE	PRICE	PRICE	PRICE
Least complex, effort intensive	JENSEN	JENSEN	JENSEN	COCOMO

The JENSEN group had the lowest mean values in three of the four variables (UTIL, RELY and TECHFTR). COCOMO projects tended to have relatively higher effort intensive values in three of the four variables (UTIL, RELY and TECHFTR). Both of these groups showed exactly the opposite tendancies for the CPLX variable. For CPLX, the projects that COCOMO most accurately estimated tended to have values reflecting relatively lower levels of intensity while and Jensen projects had determinant values relatively higher in complexity. Both SLIM and PRICE S tended to estimate projects most accurately that had determinate values relatively in the middle.

Based on the values of the variable means by groups, two discriminant function showed statistical significance. Together these two functions accounted for 98.8% of the explained variance. Further analysis will be restricted to the first two functions. Table IV shows the value of the group centroids.

TABLE IV

Group Centroids

	Func 1	Func 2
PRICE	-0.49	-0.34
COCOMO	1.32	1.47
SLIM	0.55	-0.39
JENSEN	-1.90	0.56

The group centroids show the proximity and relative position of the group means. Figure I is a plot of the group

centroids. Function 1 is on the X-axis, with Function 2 on the Y-axis.

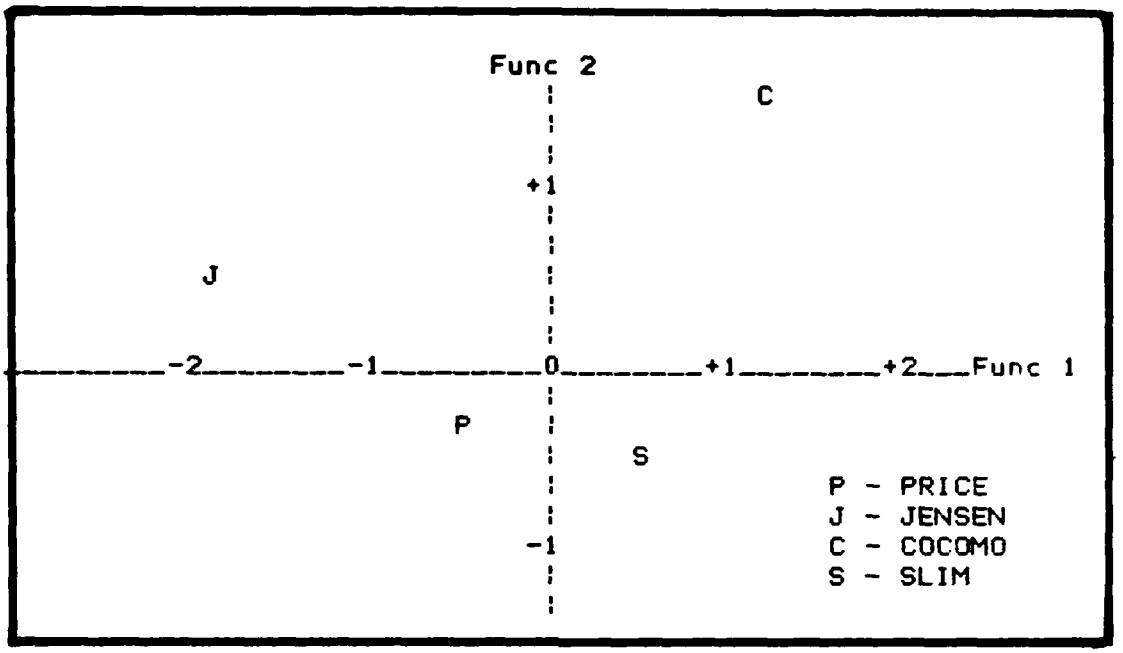


Figure 1. Plot of Group Centroids

Function 1 accounts for 69.7% of the explained variation and Function 2 accounts for 29.1% of the variation. Figure 1 reflects the proximity of PRICE and SLIM as well as the dispersion between COCOMO and the JENSEN model.

Further information can be gained by analyzing the discriminant function coefficients. The standardized coefficients reveal the relative magnitude of influence each variable has in discrimination. Table V shows the standardized and unstandardized discriminant coefficients. The absolute value of the standardized coefficients reveal the relative contribution each variable makes to the discrimination function. For example, CPLX contributes

nearly twice as much discriminating ability than any one of the other variables in both Function One and Function Two. This would indicate that the relative estimating performance of the models is most strongly influenced by the value of CPLX for the softare being estimated.

TABLE V

Canonical Discriminant Function Coefficients

	Standardized		Unstandardized	
	Func 1	Func 2	Func 1	Func 2
CPLX	2.06	-2.04	18.63	-18.46
RELY	-0.90	3.46	-3.86	14.88
UTIL	0.81	-0.99	0.54	- 0.66
TECHFTR	1.25	1.06	0.26	0.22
		(constant)	-23.35	7.00

The unstandardized coefficients are useful to calculate the group centroids from the raw data. This can be done by multiplying the unstandardized coefficients by the raw data and adding the constant.

Analysis of the Classification Function

From the discriminant function a classification function is derived which can be used to classify software projects. The discriminating performance of the four selected variables will be evaluated in two ways. First, the results of the correct classifications will be ennumerated and analyzed. And second, the relative root mean squared error (RRMSE) for

each software cost estimating model will be compared to the combined RRSME of the cases selected by the discriminant function.

The classification function with CPLX, TECHFTR, RELY and UTIL correctly classified 60% of the projects using the jackknife method. Table VI indicates how the function classified each of the cases.

Two points can be made based on the results in Table VI. First, the primary errors in classifications occurred between PRICE-SLIM substitutions. Of the the 10 incorrect classifications, only four (projects 5,7,9, and 19) were not PRICE-SLIM. This frequent substitution error can be explained by the proximity of the group centroids for those models.

The second point comes from the last column in Table V. In the ten cases which were misclassified, the actual model was not rated a strong second. In fact, in seven of the ten misclassifications, the actual model received no better than 10% probability of being the correct model. In the ten cases of misclassification, eight of the selected models provided either the worst or second worst estimates. This tendancy explains why a 60% classification score was unable to significantly decrease the overall error. Table VII displays the relative root mean square and the average percent error.

TABLE VI

Jackknifed Classification Performance

Case	Actual Model		First Choice	Prob	2nd Choice	Prob	Prob of Actual
1	SLIM		SLIM	.454	PRICE	.272	
2	SLIM		SLIM	.454	PRICE	.272	
3	SLIM		SLIM	.454	PRICE	.272	
4	SLIM		SLIM	.454	PRICE	.272	
5	COCOMO	*	SLIM	.646	PRICE	.331	.008
6	PRICE	*	SLIM	.597	COCOMO	.300	.097
7	PRICE	*	JENSEN	.999	PRICE	.001	.001
8	JENSEN		JENSEN	.893	PRICE	.103	
9	JENSEN	*	PRICE	.752	SLIM	.218	.015
10	PRICE		PRICE	.519	JENSEN	.270	
11	SLIM	*	PRICE	.665	JENSEN	.270	.065
12	PRICE		PRICE	.519	JENSEN	.270	
13	SLIM	*	PRICE	.304	JENSEN	.037	.188
14	JENSEN		JENSEN	.553	PRICE	.349	
15	JENSEN		JENSEN	.729	PRICE	.759	
16	SLIM	*	PRICE	.757	SLIM	.206	.206
17	SLIM		SLIM	.552	COCOMO	.366	
18	COCOMO		COCOMO	.913	SLIM	.066	
19	SLIM	*	COCOMO	.978	SLIM	.012	.012
20	COCOMO		COCOMO	.952	SLIM	.039	
21	SLIM		SLIM	.652	PRICE	.335	
22	PRICE	*	SLIM	.811	COCOMO	.097	.087
23	PRICE	*	SLIM	.824	PRICE	.142	.142
24	SLIM		SLIM	.530	PRICE	.424	
25	SLIM		SLIM	.530	PRICE	.424	

* indicates an incorrect classification

SUMMARY

Model Group	Percent Correct	Number of Cases Classified Into Group			
		PRICE	COCOMO	SLIM	JENSEN
PRICE	33.0	2	0	3	1
COCOMO	66.7	0	2	1	0
SLIM	66.7	3	1	8	0
JENSEN	75.0	1	0	0	3

TABLE VII

Comparison of Overall Error

	PRICE	COCOMO	SLIM	JENSEN	DISCRIM	RANDOM
RRMSE	0.95	1.36	1.01	1.26	1.03	1.06

Table VII shows that despite the 60% classification score, the discriminant function has a RRMSE larger than both PRICE S and SLIM. Furthermore, RRMSE for the expected error was only slightly greater than the RRMSE for the discriminant function. These results indicate that the use of pre-analysis determinants did not improve overall estimating capability for the 25 projects used in this study. For the data at hand, an analyst could have had less overall error by using only PRICE S or SLIM. Because the RRMSE(Random) was not significantly larger than RRMSE(Discriminant), it would be just as accurate to randomly select a model as it would to use the given discriminant function to select a model.

Because the determinants were able correctly classify a majority of the cases indicates that a statistical relationship was discovered between the values of the determinants and the model which most accurately estimated costs for that project. However, the inability to reduce overall estimating error functionally disqualifies the current application of the discriminant technique.

In retrospect, the problem appears to have been in the original project groupings. The members of each model group were assigned based on the single most accurate estimate,

irregardless of the magnitude of error. The resulting discriminant function ignored the magnitude of error in determining the group classifications as well. This point will be covered more fully in the conclusions.

V. Conclusions and Recommendations

The goal of this research effort was to improve software development estimations. Even though that goal remains unfulfilled, important knowledge has been gained. The conclusions will be drawn in light of the three research questions posed in Chapter One. Following the conclusions will be a compilation of the factors which proved to be limitations in this study. And finally, two recommendations for further study are proposed involving other applications of determinant analysis and cost estimating model selection.

Conclusions

The introduction listed three research questions which guided this investigation in pre-analysis determinants. Those questions will be addressed in turn.

Question 1. Do software development cost estimating models accuracy correlate to any specific software characteristics?

Statistically, the four models showed a trend with respect to the four determinant variables: CPLX, TECHFTR, UTIL, RELY. Sixty percent of the classifications were correct. PRICE S and SLIM tended to estimate costs that had relatively intermediate values for the determinant variables. COCOMO most accurately estimated the cost of software that tended to have relatively high values for

TECHFTR, UTIL and RELY and relatively low values for CPLX. The JENSEN groupd had relatively low values for TECHFTR, UTIL and RELY and relatively high values for CPLX.

Question 2. Which determinants best differentiate software characteristics and can they be used by analysts to select the cost model which will return the most accurate estimate?

The determinants with the most statistical discriminating power were CPLX, UTIL, RELY and TECHFTR. They were successful at selecting the appropriate model most of the time (60%).

Question 3. Will this ability to predict the most accurate cost model significantly improve software cost estimating?

Under the current application no significant benefit would result from the use of the determinants. The Relative Root Mean Square Error (RRMSE) for the predicted estimates was 1.03 and the RRMSE(Random) was 1.06. These statistics indicate that a random selection of estimating models would have produced as much error as the selection of models using pre-analysis determinants. The results found in this study are subject several limitations which may have had some undeterminable effect on the relative estimating accuracy of the respective software models.

Limitations in the Research

Although, the limitations of this study have been included in Chapter 3 and Appendix B, their ennumeration in the conclusion lends perspective both to the research results as well as the recommendations for further study. All the limitations in this study can be traced to the need for high quality, complete data on software development costs. This problem has been a significant factor in other software research efforts (12:175, 36:5-20).

The data in this study was all second hand. Access to the original software developers about questions in the data was not possible. Interaction with the actual programmers and developers can be very helpful in insuring the correct software characteristics are understood. The lack of interaction also prevented the acquiring of missing input variables. Where data was missing nominal or default values were used.

In addition to missing data, the lack of consistent development schedule information prevented equal calibration of all the models. PRICE S and SLIM were calibrated for Projects 1-12. None of the other models or projects were calibrated. The input variables for Projects 1-12 were derived for both COCOMO and Jensen based on the data available from the PRICE S and SLIM input variables for those projects.

The information for Projects 13-25 was more complete

than for the first twelve. Nevertheless, for PRICE S a default value of 3.3 was use for RESO. PRICE S also used conversion factors on projects 13-25 for both the lines of object code input and the dollars per manmonth output.

The effective study of software estimating model selection would be improved by more and higher quality data. Ideally, enough data is needed to calibrate each model for each software project estimation. Further, the ideal study would permit interaction between the software developer and analyst using the estimating models. The perfect research situation never exists, however a need for quality software estimating data does exists. The lack of comparable software data provided limitations in this study. Along with the addition of more and better data, there are other recommendations for this study that may prove effective.

Recommendations for Further Study

The results from the data showed no improvement in software cost estimating through the use of pre-analysis determinants. Important changes to the methodology used in this study could potentially improve our overall estimating ability. Two recommendations are offered for ways to improve the investigation into pre-analysis determinants.

One area of future study concerns the manner of group classifications for the software cost estimating models. In the current study, a software project was classified to the

model group that most accurately estimated its development cost. However, that type of classification does not contain any information about how accurate the best estimate was nor does it indicate how accurate the other models estimated the development effort. Some projects had more than one estimate within 10% of the actual, nevertheless that project was assigned to only one group. A form of grouping needs to be devised to acknowledge the model which came in a close second.

This problem expressed itself in another way. Some projects had no estimates within 30%. In those cases, the projects were identified as most accurate. Classifying in such an all-or-nothing manner does not properly capture the information needed to most effectively indentify determinants. Groupings which more closely represent the accuracy of each model could improve the ability to predict the most accurate cost model through pre-analysis determinants.

A second recommended research area involves inverse analysis. Rather than trying to classify projects according the cost model with the most accurate estimate, classification could be made according to which models had the worst estimates. An inverse classification need not be a relative worst estimate. The classification could be for any model whose estimate deviates by more than specified amount. This type of investigation would highlight the software characteristics which cause poor cost estimates.

Changes in the methods of classification and grouping may succeed in identifying the software characteristics which enables the selection of the best model for a particular type of software. The problem of identifying those characteristics remains to solved.

Appendix A: Summary of the Discriminant Analysis Techniques Used in this Study

To aid those not familiar with discriminant analysis, methodology background is offered on the procedures and terminology relevant to this study. Following the background information on discriminant analysis, the method of variable selection will be covered.

Background Information on Discriminant Analysis

The background in this section came for the introductory section on discriminant analysis in the Statistical Package for the Social Sciences (SPSS)(26). For a more detailed explanation the reader should refer to the SPSS.

Discriminant analysis allows two or more groups to be statistically distinguished (26:435). To do this, "the researcher selects a collection of discriminating variables that measure characteristics on which the groups are expected to differ" (26:435). These variables are called determinant variables. In this study, the groups were the software projects for which a particular estimating model produced the most accurate estimate. For example, the SLIM group contained those software projects for which SLIM estimated the development costs more accurately than any other model. The same was true for the PRICE, JENSEN and COCOMO groups.

The software cost estimating model's input variables are

theoretically the most significant cost drivers. Because of their cost estimating relationship with software, it was assumed that the input variables effectively captured the software characteristics which could potentially differentiate between one group and another. In a similar way, the relative height of a basketball player could be an effective discriminant variable in differentiating between the playing positions on a basketball team. In this research, the input variables became the determinant variables. Discriminant analysis statistically evaluates each of the determinant variables for their ability to discriminate between the groups. The determinant variables that explain the most variation between the groups are selected for inclusion in the discriminant function (26:435). The discriminant function is used to differentiate between the groups and has the form

$$D_i = d_{i1}Z_1 + d_{i2}Z_2 + \ldots + d_{ip}Z_p$$

where

D_i = the score of the ith discriminant function;

d's = the weighting coefficients;

Z's = the standardized values of the p discriminating variables.

More than one discriminant function can be formed to describe the separation between groups. The number of discriminant functions is limited by the number of groups in the analysis (26:435). The number of discriminant functions must be no more than one less than the number of groups. With four

software model groups, the maximum number of discriminant functions is three. The selected variables are standardized, weighted with coefficients and added together to form the discriminant score (D). The coefficients are chosen to minimize the differences between the D's for projects of similar groups and to maximize the differences between the D's of members of opposite groups (26:435). The discriminant functions serve as the basis for both analysis and classification (26:435).

Analysis of the discriminant function can provide theoretical insight into the similarities and differences between groups. For each group the discriminant scores can be averaged to produce the group mean for that particular function (26:443). Averaging the groups means from all the functions produces the group centroid. Close proximity between the group centroids indicates strong similarities between the members of the groups.

Each function explains some part of the discrimination (26:443). How much each function contributes to the total amount of discrimination can be determined by the percent of variation attributed to the function. The the percent of variation for all the functions sums to 100. Within each function the standardized coefficients reveal how much each variable contributes to the discriminating power of that function. Discriminant analysis will also output the unstandardized coefficients (26:443). The unstandardized coefficients can be multiplied by the values of the

respective discriminant variables and summed together. To adjust for the grand means a constant is added, resulting in the group mean for that function (26:443). Just as the discriminant function provides basis to analyze the data, it also provides the basis to classify the data.

Classification takes place after selecting the variables which satisfactorily discriminate between cases of known group memberships. How the variables are selected for the discriminant function will be covered in the next section. Once the variables have been selected for the discriminant function, a classification function can be produced which will classify cases into groups based on the values of their discriminant values (26:445). Thus, the classification function can be used to check the adequacy of the original discrimination function and also to classify cases of unknown group membership (26:436).

How accurately the classification function can assign each case to a group indicates the strength or adequacy of the function (26:436). Unfortunately, classifying the same data used to derive the function will produce a biased classification score. The bias comes from the fact that the discriminating function contains information about the specific case it is attempting to classify. The problem of biased classification scores can be reduced by a process called jackknifing (9:711).

Jackknifing involves removing one case at a time from the data set and calculating the discriminant and

classification functions based on the remaining cases (9:731). The classification functions are used to classify the removed case. The process is repeated producing a different discriminant function for every case in the data the. Similarity between all of the discriminant functions can be assumed if the function appears stable. The stability of the discriminant functions is indicated by the difference between the unjackknifed (biased) classification accuracy scores and the jackknifed (unbiased) classification accuracy scores. Where significant differences exist between biased and unbiased scores, the discriminant functions should be considered unstable. The instability of the classification function usually results from the discriminant function chasing random error in its attempt to differentiate between the groups. In those cases, the discriminating function changes erratically as individual cases are removed from the data. Small differences between biased and unbiased scores indicate a stability on the original discriminant function.

With a stable, accurate classification function cases of unknown membership can be classified based on the values of the discriminant variables (26:445). The procedure for classification involves the sum of the weighted discriminated variables plus a constant. Each group has its own classification function with unique weighting coefficients and a unique constant. According to the SPSS manual (26:445) "Under the assumption of a multivariate normal distribution, the [classification function sum] can be converted into

probabilities of group membership." Thus, the group with the highest classification function sum is the group in which the given case has the greatest probability of membership. Ideally, a discriminant function should be validated by using it to classify a new set of known cases. Unfortunately, the shortage of available data will limit this study to jackknifing the set of twenty-five projects or cases to validate the functions.

Selection Criteria for the Discriminant Variables

With a brief background in how discriminant analysis functions it will be easier to understand the method employed to select the variables included in the discriminant function. For this study the Stepwise Method in the Bio-Medical Decision Package (BMDP) was used. The Stepwise Method begins with no variables in the discriminant function and then evaluates all the variables ability to discriminate between the groups. The discriminating power of the variables is determined by the ability of the variable when included in the equation to increase the value of a statistic called Mahalanobis D. Mahalanobis D reflects the distance between the two closest groups. The relative increase of Mahalanobis D attributable to each variable is expressed in an F-to-enter value. The larger the F-to-enter value for a respective variable the more discriminating ability. The first variable to be included in the equation is the one with

the largest F-to-enter value. All of the remaining variables are then evaluated for their ability to maximize Mahalanobis D, given the other variable is already in the equation. As new variables are added the variables already in the equation have their F value evaluated. If the new added variable causes the F value of a previous included variable to drop below a specified level, that variable will be removed from the function. In this study, some variables entering the discriminant equation were manually controlled. The controlled variables were primarily default values used because of incomplete data, discriminating off default variables would have no real meaning.

D, given the other variable is already in the equation. As new variables are added the variables already in the equation have their F value evaluated. If the new added variable causes the F value of a previous included variable to drop below a specified level, that variable will be removed from the function. In this study, some variables entering the discriminant equation were manually controlled. The controlled variables were primarily default values used because of incomplete data, discriminating off default variables would have no real meaning.

Appendix B: Model Descriptions

Description of the PRICE S Model

PRICE Systems Division of RCA, Cherry Hill, New Jersey developed and maintain PRICE S and is only one of an entire family of RCA parameteric cost estimating models. The acronym PRICE stands for Programmed Review of Information for Costing and Evaluation. PRICE S, the software component of the PRICE models estimates costs by systematically adjusting an initial estimate based on the size and description of the project software and development environment. The methodology does not try to fit historical data and into hypothesized relationships in a linear regression manner. Instead, PRICE S is based on the opinions of knowledgable software development managers collected and correlated in a delphi method. These expert opinions about perceived relationships between between costs and software development techniques were collected and incoded into alogorithms by Frank Freiman, the creator of PRICE S. Due to the proprietary status of PRICE S, the exact nature of the applied algorithms is not public knowledge.

PRICE S Input Variables

The total number of possible inputs for PRICE S includes 64 variables, most of which contain default values if not known. The minimum inputs needed for an estimate is eight. The variables known, and used for this research project follow.

INST - The size of the software program is inputed in the total number of delivered, executable, machine level instructions. This differs from the other four models which use source code. Executable machine-level instructions are the most basic operations of the computer. Source code are the lines of code as written by the programmer. Source code are processed by a language compiler resulting in machine executable code. Generally, one line of source code will produce more than one line of machine code.

APPL - measures the difficulty of the program application. Each section of the program is assessed a weighting of difficulty ranging from 0.865 to 10.95 and multiplied by the percent that the section occupies of the entire program. Values for APPL range from 0.865 for a program that is 100% simple mathematical application, to 10.95 for a program that is 100% on-line operations.

RESO - represents the overhead and labor rate for the computer developers/programmers used to develop the software. It reflects the individual software development company's productivity quotient. Defense Industrial values

typically range from 3.3 to 3.4.

UTIL - is the fraction of available memory capacity and or timing constraints . Extra design and programming effort is required to adapt a program within a restricted computer memory or timing constraints.

PLTFM - is the level of reliability and testing documentation needed. PLTFM varies from 0.6 for internally developed production center software to unmanned and manned spacecraft software at 2.0 and 2.5 respectively.

CPLX - attempts to capture the numerous extenuating circumstances surrounding the development environment. Its nominal value is one, indicating average programmers working on a average program. Depending on the type and number of complicating circumstances such as changing requirements or new language, CPLX can range from 0.6 to 1.8.

NEWD,NEWC - represent the percent of the project that is new in design and that requires new code, respectively. A zero value represents for each of these variables indicates that no effort is required, there is no new designing or new coding needed. The top value is 100%, indicating a completely new project requiring a complete programming effort.

The data base contained all the actual values for these variables except cases 13-25 did not have RESO. The defense industry standard of 3.3 was used for the RESO value for those cases.

PRICE S Output Used for Comparison

PRICE S provides output for development cost in dollars. To provide comparability with the other models dollars were converted to man-months. Accoding to Mr. Earl King, Manager in Customer Relations at RCA PRICE, for the defense industry one man-year of software development equates to $105,000. The outputs for PRICE S in dollars were were converted to man-months using the defense industry average. The phases included in the development effort for all twenty-five projects were: product design, detail design, code and unit test and finally test and integration.

Information on PRICE S came from the reference manual of the model (29).

Description of the COCOMO Model

The Constructive Cost Model (COCOMO) is a non-proprietary model developed by Barry W. Boehm and illustrated in detail in his software text book Software.Engineering Economics. The model uses an historical data base of 63 projects and using linear regression fit a function through the data to try explain the relationship between the lines of source code and the development effort. COCOMO operates in three modes (organic semi-detached, and embedded) depending on the type of software being estimated. Each mode has a unique equation.

The organic mode is characterized by a stable development environment, with minimal concurrent development of related hardware. Simple design requirements for interface and data processing along with a low priority on completion schedule further reflect the organic mode. The organic mode correlates to high productivity and less diseconomies of scale. The development effort equation for the organic mode is

$$MM = 3.2\ KDSI^{1.05} \prod EM_i \qquad (2)$$

where

MM = product development effort;

KDSI = effective software sizing, in thousands of source instructions;

EM = the 15 COCOMO development effort multipliers.

In contrast to the organic mode, the embedded mode is characterized by tight constraints. Embedded software must operate within a complex of hardware, regulations, and operational procedures. In short, the software is expected to conform to the enviroment specifications. Embedded software typically is expected to absorb more changes and requries a higher level of development effort. The software equation for the embedded mode is

$$MM = 2.8\ KDSI^{1.2} \prod EM_i. \qquad (3)$$

Between the organic and embedded modes lies the semi-detached. It occurs when blend of the characteristics of the organic and embedded exists. The semi-detached mode software equation is

$$MM = 3.0\ KDSI^{1.12} \prod EM_i. \qquad (4)$$

Input Variables of COCOMO

Each of the three operating modes use the same input variables as the development effort multipliers. The possible ratings for each variable are (1) very low, (2) low, (3) nominal, (4) high, (5) very high, and (6) very high. Each variable has at least four of the possible ratings available, but in some cases the (1) very low, (2) low and (6) extra high rating are not appropriate responses for a

particular variable, in those instances (1), (2)and (6) are not included as a rating response. For every variable Boehm furnishes exact parameters to objectively determine the value of the ratings. And each of these ratings has a specific weighting which is used as a multiplier in the software equation. A description of the cost drivers follows.

SIZE - is measured in thousands of delivered source instructions.

DATA - the size of the project data base.

CPLX - the level of complexity of the software being developed.

TIME - the degree of execution time constraint on the target computer, that is, the computer on which the software is intended to operate.

STOR - the percent of main storage used on the target computer by the software being developed.

VIRT - the degree that the virtual machine changes during development. The virtual machine is the complex of hardware and software used by the software to accomplish its task.

TURN - the amount of time between input and hard copy turn around for the development computer. The development computer is the system on which the software is being developed.

ACAP - the analysts capability with respect the industry average.

AEXP - the amount of experience measure in years that the analysts have had in the area of application of the software being developed.

PCAP - the capability of the programmers relative the industry average.

VEXP - previous experience in years of the programmers with the virtual machine.

LEXP - number of years the programmers have been using the language to used on the software being developed.

MODP - degree that modern programming practices are being

applied to the development effort.

TOOL - the sophistication and efficiency of the software development tools used in the effort.

SCED - the amount of compression or stretchout exerted on the completion schedule. COCOMO differs from both SLIM and JENSEN in its concept of completion schedule. COCOMO proposes an optimal schedule time and penalizes for either program slippages or compressions. Both SLIM and JENSEN estimate a lower effort requirement for projects developed in longer periods of time.

The estimates used in the analysis for comparison came from a detailed version of COCOMO. The detailed version of COCOMO adjusts values for the effort multipliers according to each of the development phases and produces slightly more accurate estimates than the intermediate version. The discriminant analysis used the effort multiplier values from the intermediate version of COCOMO. The intermediate version assumes the effort multiplier for each development phase remains constant. Discriminant analysis required a single value be used per cost driver, for each project. The intermediatate multipliers closely approximate the average value of the detailed multipliers for each cost driver. The single multiplier values from the intermediate served as a useful proxy for the phase adjusted multiplier values used in the detailed version.

Table VIII indicates the values used in the discriminant analysis for each respective effort multipier rating. The values listed in Table VIII come from the intermediate version of COCOMO.

TABLE VIII

COCOMO Cost Driver Effort Multipliers

Cost Drivers	Effort Multiplier Ratings					
	Very Low	Low	Nominal	High	Very High	Extra High
RELY	0.75	0.88	1.00	1.15	1.40	
DATA		0.94	1.00	1.08	1.16	
CPLX	0.70	0.85	1.00	1.15	1.30	1.65
TIME			1.00	1.11	1.30	1.66
STOR			1.00	1.06	1.21	1.56
VIRT		0.87	1.00	1.15	1.30	
TURN		0.87	1.00	1.07	1.15	
ACAP	1.46	1.19	1.00	0.86	0.71	
AEXP	1.29	1.13	1.00	0.91	0.82	
PCAP	1.42	1.17	1.00	0.86	0.70	
VEXP	1.21	1.10	1.00	0.90		
LEXP	1.14	1.07	1.00	0.95		
MODP	1.24	1.10	1.00	0.91	0.82	
TOOL	1.24	1.10	1.00	0.91	0.83	
SCED	1.23	2.08	1.00	1.04	1.10	

COCOMO Output Used For Comparison

COCOMO includes in its definition of software development the same phases used by PRICE. Those phases include: product design, detailed design, code and unit test, and integration and test. For the first thirteen projects, the data included actual development effort up through code and unit test. Therefore, for the first thirteen projects, the COCOMO estimates were adjusted to exclude the integration and test phase. In each case, the same phases included in the COCOMO estimate were also included in the actual figures for development effort.

A sizeable problem existed in this research effort concerning data. For comparison, input values and actual development effort data were needed for all four models simultaneously. The first twelve projects had only PRICE S and SLIM inputs. To obtain COCOMO input variables for the first twelve projects it was necessary to translate the PRICE S inputs. Despite this unfortunate necessity, all of COCOMO's best estimating performances came out of the first twelve projects.

Description of the SLIM Model

Quantitative Software Management, Inc. of McLean Virginia has proprietary control of SLIM (Software Life Cycle Model). Lawrence H. Putnam developed SLIM from based on the hypothesis of P. V. Norden (Norden:71-101) that software development effort should follow the Rayleigh distribution function for learning. Putnam applied the Rayleigh learning principle equation to software development resulting in the following equation:

$$Ss = C_k K^{1/3} Td^{4/3} \qquad (6)$$

where

Ss = number of delivered source instructions
K = life-cycle effort in man-years
C_k = the Technology Constant, known for the given environment;
Td = the development time in years.

SLIM allows C_k to be calibrated using past projects, or using a description of the project, SLIM can also calculate C_k for you. Because of the Rayleigh curve orientation, SLIM can readily perform effort-schedule trade-offs. Equation 5 reveals that SLIM suggests a radical trade-off between effort and schedule. Putnam refers to this as the fourth power trade-off law because a unit increase in schedule can reduce to the development effort by the power of four.

SLIM Input Variables

SLIM has only three primary input variables SIZE, LEVEL or Difficulty Gradient, and Technology Factor.

SIZE is measured in lines of source code, that is the lines of code actually written by the programmer before being compiled.

LEVEL represents both the difficulty of the program and the staffing technique used on the project. Lower values LEVEL equate to new programs with highly intricate interfaces, these programs generally require low staffing levels to reduce the intercommunication and coordination load during development. Higher values of LEVEL reflect larger staffing levels because these projects are merely composites of smaller sections, and communication problems are not as much of a negative factor during development. LEVEL can assume discrete values depending on the following considerations:

(1) The system is entirely new - designed from and coded from scratch. It has many interfaces and must interact with other systems within a total management information system structure.

(2) This is a new stand-alone system. It is also designed and coded from scratch but is simpler because the interface problem with other systems in eliminated.

(3) This is a rebuilt system where large segments of existing logic exist. The primary tasks are recoding, integration, interfacing and minior enhancements.

(4) This is a composite system made up of a set of independent subsystems with few interactions and interfaces among them. Development of the independent

subsystems will occur with considerable overlap.

(5) This is a composite system made up of a set of independent subsystems with a minimum of interactions and interfaces among them. Development of the independent subsystems will occur virtually in parallel.

Technology Factor for most systems ranges from 4 to 15. The lower values reflect more advanced applications, while the larger values describe simpler systems. The first twelve projects in the data for this research project had technology factors calibrated from historical data. The last thirteen projects used the technology factor calculated by SLIM, IBM-PC version, based on inputs describing the project. Those input variables follow.

MONTH, YEAR - the month and year the detailed design started

ONLINE - the percent that the development that will be online, interactive mode.

COMPUTER AVAILABILITY - the percent the development computer will be dedicated to this project.

PRIMARY LANGUAGE - name of the computer languaged used.

SECONDARY LANGUAGE - self-explanatory.

ASSEMBLY, HOL - percent of assembly, and higher order language (HOL) used.

4th GENERATION, DBMS, REPORT WRITER, SCREEN WRITER - percent usage of these development tools.

TYPE - type of application the software will be used on.

REAL TIME CODE - percent of the software requiring real time constraints.

MEMORY - percent of the target computer memory to be used by the software.

INTERFACES - low, moderate, or high degree of interface complexity.

NEW DESIGN - percent of new design needed for the project.

MANPOWER AVAILABLITY - percent of total manpower available for this project.

STRUCTURED PROGRAMMING, TOP-DOWN DESIGN, DESIGN/CODE WALKTHROUGHS, PROGRAM LIBRARIAN - low, medium, or high degree of usage for these modern programming practices.

PERSONEL EXPERIENCE: OVERALL, LANGUAGE, SIZE AND APPLICATION, MANAGEMENT, DEVELOPMENT COMPUTER - low, average or extensive experience in the listed items.

SLIM Output Used for Comparison

SLIM normally defines development effort as beginning with detailed design and ending with a fully operationally system. The first twelve projects compared estimates and actuals that included SLIM's normal definition. Due to data limitations, projects 13 through 25 compared estimates and actuals that included the front-end phase called product design. In each case, the phases included in the actual figures were also included in the estimate figures.

In addition to the adjustments in total development effort, an additional input was made for projects 13 through 21 not asked for by the model. The input, peak staff loading, was necessary due to SLIM's sensitivity to schedule and effort trade-offs (the fourth power trade-off law). Because the schedule data incomplete, it was not possible to determine if the projects were incrementally developed. Without schedule information, SLIM only returns the minimum schedule time which is the maximum development effort. SLIM was able to adjust the schedule away from the maximum

development effort with the use of the peak staff loading input. On projects 22 through 25, the maximum development effort was assumed because of no objective ability to adjust the estimate otherwise.

Information on SLIM came from the user's manual of the IBM PC version of the model (28) and a summary of the model given in the Thibodeau report (36:A63-A80).

Description of the Jensen Model

Computer Economics Incorporated (CEI) has proprietary control of the Jensen software development model. Randall W. Jensen developed the model incorporating the earlier work of Putnam (27) and Doty Associates (15). Similar to the SLIM model, the Jensen bases its estimate on the Rayleigh development curve. The software equation used in the Jensen Model is

$$S_e = C_{te}\ t_d\ K^{1/2} \qquad (6)$$

where

S_e = the effective software size in lines of source code including the impact of modifications to existing software;

C_{te} = the effective developer constant;

t_d = the development time;

K = the total life cycle effort in person-years.

The effective developer constant C_{te} is derived from another constant called the basic technology constant C_{tb}. The basic technology constant reflects the basic development capability of the developer such as the capability and experience of the organization, the use of modern development practices and the use of software tooling. The basic technology factor is then adjusted by thirteen environmental factors which include project reliability requirements, memory constraints, system complexity, implementation hardware and language etc. The equation describing the

relationship between the basic technology factor and the effective technology factor is

$$C_{te} = C_{tb} / \left(\prod f_i \right) \qquad (7)$$

where

C_{tb} = the basic technology factor;

f_i = the ith environmental adjustment factor.

Jensen Input Variables

The Jensen Model had the most number of input variables of all the models. With the exception of task complexity, the input variables have rating options similar to COCOMO. Each variable may be rated (1) very low, (2) low, (3) nominal, (4) high, (5) very high and (6) extra high. And also like COCOMO, for some variables the upper and lower ratings are not appropriate and are not available options for those particular variables. The task complexity variable resembles the SLIM variable technology factor with options ranging from 4 to 28 describing the complexity of the task being accomplished by the software. The other variables follow.

SIZE - measured in lines of delivered source including adjustments for modifications of existing software.

ACAP - the analysts capability with respect the industry average.

AEXP - the amount of experience measure in years that the analysts have had in the area of application of the software being developed.

PCAP - the capability of the programmers relative the industry average.

TURN - the amount of time between input and hard copy turn around for the development computer. The development computer is the system on which the software is being developed.

TOOL - the use of automated software development tools used in the project.

MODP - the use of modern development practices.

APPL - The number of years necessary for a programmer to become nominally proficient in the specific applications area in which the software will operate.

DISP - complexity of the display interface requirements.

HOST - amount of effort needed to adapt the software for use in the target computer.

LEXP - number of years the programmers have been using the language to used on the software being developed.

LOSS - effective loss due to software failure or the degree of reliability required of the system.

MEMC - the amount of design consideration necessary to fit software program into the main storage memory of the target computer.

MULT - number of sites at which the development took place.

RCHG - the frequency and severity of any requirement changes.

RDED - the percent of the resources dedicated to the software development.

RLOC - location of the support resource relative to the development location.

RTIM - percent of the program operating under real time opertion.

SEXP - previous experience in years of the programmers with the virtual machine.

TIMC - the degree of execution time constraint on the target computer, that is, the computer on which the software is intended to operate.

VMVL - the degree that the virtual machine changes during development. The virtual machine is the complex of hardware

and software used by the software to accomplish its task.

LANG - the complexity inherent to the language used in the software program.

SYST - the complexity of the development computer system.

Jensen Output Used for Comparison

The Jensen Model defines development as beginning with System Design Review (SDR) and ending with the Final Qualification Test (FQT). This definition for development was used for all twenty-five Jensen efforts.

Similar to SLIM, the Jensen Model returns the maximum development effort (i.e. the minimum schedule) if not schedule constraint is offered. As with SLIM, for cases 13 through 21, an additional input was made for the Jensen Model. The peak manloading was inputed to prevent a maximum effort estimate. For cases 22 through 25, no staffing levels were available, so a minimum schedule was assumed.

Information on the Jensen model came from the reference manual for the JS-2 version of the model.

Appendix C: Model Inputs

PRICE S Inputs

Case	INST	APPL	RESO	UTIL	PLTFM	CPLX	NEWD	NEWC
1	150800	5.71	2.25	0.80	1.20	0.60	0.20	0.30
2	101800	5.73	2.25	0.80	1.20	0.60	0.90	0.90
3	101800	5.65	2.25	0.80	1.20	0.60	0.60	0.60
4	110400	5.52	2.25	0.80	1.20	0.60	0.70	0.80
5	223800	5.55	2.25	0.80	1.20	0.60	0.70	0.80
6	65600	5.58	2.25	0.80	1.20	0.60	0.50	0.60
7	39120	2.51	2.13	0.50	1.00	1.00	1.00	1.00
8	77470	2.51	2.13	0.50	1.00	1.00	1.00	1.00
9	142732	3.00	1.88	0.50	1.00	1.00	0.78	0.83
10	148700	3.00	1.88	0.50	1.00	1.00	0.90	0.50
11	194748	3.00	1.88	0.50	1.00	1.00	0.90	1.00
12	585200	3.00	1.88	0.50	1.00	1.00	0.85	0.95
13	91700	8.09	3.30	0.60	1.20	1.50	1.00	1.00
14	51958	4.99	3.30	0.60	1.20	1.10	1.00	1.00
15	182068	6.35	3.30	0.80	1.20	1.00	0.60	0.60
16	40462	8.90	3.30	0.90	1.70	1.10	1.00	1.00
17	63944	5.08	3.30	0.50	1.20	1.00	1.00	1.00
18	31277	5.98	3.30	0.60	1.40	1.00	0.34	0.34
19	32696	5.98	3.30	0.60	1.40	1.00	0.13	0.13
20	47525	5.98	3.30	0.60	1.40	1.00	0.34	0.34
21	29250	3.00	3.30	0.50	1.40	1.10	0.40	0.40
22	82500	8.46	3.30	0.80	1.40	1.00	1.00	1.00
23	83050	8.00	3.30	0.80	1.80	1.00	1.00	1.00
24	66000	8.00	3.30	0.80	1.80	1.00	1.00	1.00
25	81900	8.00	3.30	0.80	1.80	1.00	1.00	1.00

COCOMO Inputs

Case	SIZE	RELY	DATA	CPLX	TIME	STOR	VIRT	TURN	ACAP	AEXP	PCAP
1	13400	H	N	H	H	H	N	N	VH	VH	VH
2	25000	H	N	H	H	H	N	N	VH	VH	VH
3	26800	H	N	H	H	H	N	N	VH	VH	VH
4	23750	H	N	H	H	H	N	N	VH	VH	VH
5	52350	H	N	H	H	H	N	N	VH	VH	VH
6	13250	H	N	H	H	H	N	N	VH	VH	VH
7	12381	L	N	N	N	N	N	N	N	N	N
8	27519	L	N	N	N	N	N	N	N	N	N
9	32276	VL	N	N	N	N	N	N	H	H	N
10	20534	VL	N	N	N	N	N	N	H	H	N
11	46253	VL	N	N	N	N	N	N	H	H	N
12	127775	VL	N	N	N	N	N	N	H	H	N
13	26200	VH	L	VH	N	H	L	L	N	VL	N
14	15987	H	N	H	H	H	VL	L	VH	VL	N
15	56021	VH	L	H	VH	VH	VL	L	VH	VL	VH
16	21296	H	L	H	VH	EH	VL	N	VH	H	VH
17	63544	H	N	VH	N	N	L	L	VH	N	VH
18	31277	VH	L	VH	VH	H	VL	L	VH	N	VH
19	32696	VH	L	VH	VH	VH	VL	L	VH	N	VH
20	47525	VH	L	VH	VH	VH	VL	L	VH	N	VH
21	9000	H	N	VH	N	N	VL	N	VH	N	VH
22	15000	VH	L	VH	VH	VH	VL	H	H	H	H
23	15100	VH	L	VH	VH	VH	VL	H	H	H	H
24	12000	VH	L	VH	VH	VH	VL	H	H	H	H
25	14900	VH	L	EH	VH	VH	VL	H	H	H	H

COCOMO Inputs, continued

Cases	VEXP	LEXP	MODP	TOOL	SCED	MODE	ND	NC	NI
1	N	N	N	N	N	E	20	30	30
2	N	N	N	N	N	E	90	90	90
3	N	N	N	N	N	E	60	70	70
4	N	N	N	N	N	E	70	80	80
5	N	N	N	N	N	E	70	80	80
6	N	N	N	N	N	E	50	60	60
7	N	N	N	N	N	E	100	100	100
8	N	N	N	N	N	E	100	100	100
9	N	H	H	H	N	O	78	83	83
10	N	H	H	H	N	O	90	50	90
11	N	H	H	H	N	O	90	90	100
12	N	H	H	H	N	O	85	95	95
13	VL	VL	N	H	N	E	100	100	100
14	N	H	H	N	H	E	100	100	100
15	H	N	H	VH	N	E	47	47	75
16	N	VL	H	N	N	E	100	100	100
17	H	H	N	VL	L	E	100	100	100
18	H	H	H	H	N	E	29	29	100
19	H	H	H	H	N	E	09	09	100
20	H	H	H	H	N	E	04	04	100
21	L	N	N	N	N	E	40	04	100
22	H	H	H	H	H	E	100	100	100
23	H	H	H	H	H	E	100	100	100
24	H	H	H	H	H	E	100	100	100
25	H	H	H	H	H	E	100	100	100

SLIM Inputs

Case	INST	LEVEL	TECHFTR
1	13400	5	11
2	25000	3	11
3	26800	3	11
4	23750	4	11
5	52350	3	11
6	13250	5	11
7	12381	2	9
8	27519	2	9
9	32276	3	15
10	20534	4	15
11	46253	1	15
12	127775	2	15
13	26200	1	4
14	15987	2	6
15	56021	2	3
16	21296	1	4
17	63944	4	11
18	31277	5	10
19	32696	5	10
20	47525	5	11
21	9000	3	7
22	15000	1	4
23	15100	1	1
24	12000	1	1
25	14900	1	1

Jensen Inputs

Case	EFFCT SIZE	MEMC	MULT	TCHG	RDED	RLOC	RTIM	SEXP	TIMC	VMVL	LANG
1	6767	H	N	N	N	N	H	N	N	N	N
2	23375	H	N	N	N	N	H	N	N	N	N
3	20502	H	N	N	N	N	H	N	N	N	N
4	19754	H	N	N	N	N	H	N	N	N	N
5	44322	H	N	N	N	N	H	N	N	N	N
6	9310	H	N	N	N	N	H	N	N	N	N
7	12400	N	N	N	N	N	N	N	L	N	N
8	27500	N	N	N	N	N	N	N	L	N	N
9	26163	N	N	N	N	N	N	N	L	N	N
10	18450	N	N	N	N	N	N	N	L	N	N
11	44448	N	N	N	N	N	N	N	L	N	N
12	116275	N	N	N	N	N	N	N	L	N	N
13	26200	VH	N	N	N	N	VH	VL	N	L	N
14	15987	VH	N	N	N	N	L	N	H	L	L
15	26863	H	N	N	N	N	EH	L	N	L	L
16	21296	H	N	N	N	N	N	N	N	L	L
17	63944	EH	N	N	N	N	L	H	VH	L	L
18	18289	N	N	N	H	N	H	H	N	L	L
19	13356	N	N	N	H	N	H	H	N	L	L
20	27754	N	N	N	H	N	H	H	N	L	L
21	5490	VH	N	N	N	N	N	L	N	L	L
22	15000	VH	N	VH	EH	N	VH	H	VH	L	L
23	15100	VH	N	H	EH	N	VH	H	VH	L	L
24	12000	VH	N	H	EH	N	VH	H	VH	L	L
25	14900	VH	N	H	EH	N	VH	H	VH	L	L

Jensen Inputs, continued

Case	SYST	ACAP	AEXP	APPL	MODP	PCAP	TOOL	TURN	DISP	HOST	LEXP
1	N	H	H	N	H	VH	N	N	N	N	N
2	N	H	H	N	H	VH	N	N	N	N	N
3	N	H	H	N	H	VH	N	N	N	N	N
4	N	H	H	N	H	VH	N	N	N	N	N
5	N	H	H	N	H	VH	N	N	N	N	N
6	N	H	H	N	H	VH	N	N	N	N	N
7	N	N	N	L	N	N	N	N	N	N	N
8	N	N	N	L	N	N	N	N	N	N	N
9	N	H	H	L	H	H	H	N	N	N	N
10	N	H	H	L	H	H	H	N	N	N	N
11	N	H	H	L	H	H	H	N	N	N	N
12	N	H	H	L	H	H	H	N	N	N	N
13	N	N	N	VL	N	N	H	L	H	N	N
14	N	H	VL	H	N	N	H	N	H	N	L
15	N	H	VL	N	H	H	VH	VL	N	VH	N
16	N	H	H	N	N	H	L	VL	VH	N	N
17	N	H	H	N	N	H	L	L	H	N	L
18	N	H	H	H	H	H	H	L	H	VH	L
19	N	H	H	H	H	H	H	L	H	VH	L
20	N	H	H	H	H	H	H	L	H	VH	L
21	N	H	N	H	N	H	N	N	VH	VH	L
22	L	H	H	H	H	H	H	H	N	N	H
23	L	H	H	H	H	H	H	H	N	N	H
24	L	H	H	H	H	H	H	H	N	N	H
25	L	H	H	H	H	H	H	H	N	N	H

Jensen Inputs, continued

Case	LOSS	CPLX
1	H	11
2	H	11
3	H	11
4	H	11
5	H	11
6	H	11
7	L	15
8	L	15
9	L	21
10	L	21
11	L	21
12	L	21
13	VH	8
14	H	11
15	VH	11
16	H	8
17	H	11
18	VH	11
19	VH	11
20	VH	11
21	H	15
22	VH	4
23	VH	4
24	VH	4
25	VH	4

Appendix D: Model Outputs

PRICE S Outputs

Case	EST	ACT	Absolute Deviation
1	48.0	39.6	6.4
2	85.0	79.0	6.0
3	96.0	90.7	5.3
4	78.0	95.9	18.0
5	150.0	115.7	34.3
6	39.0	28.5	10.5
7	19.0	14.5	4.5
8	35.0	129.5	94.5
9	48.0	87.2	39.2
10	38.8	44.0	5.2
11	37.2	71.0	33.8
12	176.0	192.0	16.0
13	358.3	309.0	49.3
14	129.5	62.0	67.5
15	221.7	151.0	70.7
16	271.0	197.0	74.0
17	477.4	317.0	160.4
18	58.5	46.0	12.5
19	28.2	11.0	17.2
20	88.0	59.0	29.0
21	34.3	48.0	13.7
22	709.1	1000.0	290.9
23	854.0	554.0	300.0
24	674.5	337.0	337.5
25	842.4	268.0	574.2

RRMSE = 0.95

COCOMO Outputs

Case	EST	ACT	Absolute Deviation
1	7.0	39.6	32.6
2	54.2	79.0	24.8
3	61.3	90.7	29.4
4	38.4	95.9	57.5
5	101.0	115.7	14.7
6	13.3	28.5	15.2
7	38.8	14.5	24.3
8	100.8	129.5	28.7
9	34.7	87.2	52.5
10	20.7	44.0	23.3
11	60.8	71.0	10.2
12	166.0	192.0	26.0
13	312.8	342.0	29.2
14	48.2	62.0	13.8
15	138.0	154.0	16.0
16	111.3	241.0	129.7
17	235.4	490.0	254.6
18	53.0	56.0	3.0
19	43.5	12.5	31.5
20	100.0	72.0	28.0
21	11.8	59.0	47.2
22	77.9	1219.0	1141.1
23	78.5	675.0	596.5
24	59.6	411.0	351.4
25	98.0	327.0	228.0

RRMSE = 1.36

SLIM Outputs

Case	EST	ACT	Absolute Deviation
1	45.1	39.6	5.5
2	76.9	79.0	2.1
3	90.0	90.7	0.7
4	102.2	95.9	6.1
5	307.9	115.7	192.2
6	44.6	28.5	16.1
7	27.1	17.4	9.7
8	125.4	155.6	30.2
9	38.5	71.0	32.5
10	20.7	38.0	17.3
11	34.0	36.0	2.0
12	212.1	184.0	28.1
13	354.0	342.0	12.0
14	83.0	62.0	11.0
15	140.0	154.0	14.0
16	234.0	241.0	7.0
17	509.0	490.0	19.0
18	29.7	56.0	26.3
19	15.0	12.5	2.5
20	33.7	72.0	38.3
21	55.3	59.0	3.7
22	300.0	1219.0	919.0
23	302.0	675.0	373.0
24	238.0	411.0	173.0
25	298.0	327.0	29.0

RRMSE = 1.01

Jensen Outputs

Case	EST	ACT	Absolute Deviation
1	30.4	39.6	9.2
2	134.5	79.0	55.5
3	115.0	90.7	24.3
4	110.0	95.9	14.1
5	289.0	115.7	173.3
6	45.0	28.5	16.0
7	59.0	14.5	44.5
8	153.0	129.5	23.5
9	79.2	87.2	8.0
10	62.0	44.0	18.0
11	177.0	71.0	106.0
12	559.0	71.0	106.0
13	279.5	309.0	29.5
14	65.7	62.0	3.7
15	156.5	151.0	5.5
16	204.4	197.0	7.4
17	241.2	317.0	75.8
18	153.8	46.0	107.8
19	88.3	11.0	77.3
20	212.3	59.0	153.3
21	59.8	48.0	11.8
22	146.5	1000.0	853.5
23	147.6	554.0	406.4
24	112.1	337.0	224.9
25	123.1	268.0	144.9

RRMSE = 1.26

Bibliography

1. Apgar, Henry. "Software Life Cycle Cost,"Proceedings National Estimating Society Conference. 23 June 1983.

2. Bailey, John W. and Victor R. Basili. "A Meta-Model For Software Development Resource Expenditures," National Aerospace and Electronics Conference. 107-116. IEEE Press, New York, 1981.

3. Boehm, B.W. Software Engineering Economics. Englewood Cliffs NJ: Prentice Hall, 1981.

4. Bourdon, Gerald A. and Joseph A. Duquette. A Computerized Model for Estimating Software Life Cycle Costs. Report No. ESD-TR-77-235-VOL-I, Electronic System Division, Hanscom AFB MA, April 1978 (AD-A053 937).

5. Computer Economics Inc. JS-2 Reference Manual. CEI, Marina Del Rey CA, May 1984.

6. Dean, Joseph. Software Cost Estimating Analyst, Electronic System Division. Telephone Interview. Hanscom AFB MA. 6 February 1984.

7. Devenny, Captain Thomas J. An Exploratory Study of Software Cost Estimating at ESD, MS Thesis, GSM/SM/765-4. School of Engineering, Air Force Institute of Technology. (AU), Wright-Patterson AFB OH, July 1976 (AD-A030 162).

8. Dircks, Henry F. "SOFCOST," National Aerospace and. Electronics Conference. 674-683. IEEE Press, New York, 1981.

9. Dixon, W. J., Brown M.B. Bio-Medical Decision Package. Berkeley: University Of California Press, 1979.

10. Dumas, R.L. Software Acquistion Resource Expenditure (SARE) Data Collection Methodology. The Mitre Corporation, ESD-TR-83-214, December 1983.

11. Duquette, Joseph A. Air Staff Cost Analyst. Telephone Interview. Pentagon, Washington DC. 18 January 1984.

12. Ferens, Daniel V. "Cost Models: The Essential Ingredient," National Aerospace and Electronics Conference. 175. IEEE Press, New York, 1981.

13. Gibson, Jack, Chief Cost Analysis Research (ASD/ACCR). Aeronautical System Division. Personal Interview. HQ ASD, Wright-Patterson AFB OH. 19 December 1983.

14. Hall, David M. "Making Sure We Can Lift the Sword," National Aerospace and Electronic Conference. 1018. IEEE Press, New York, 1983.

15. Herd, J.H. Software Cost Estimation Study, Vol I & II: Guidlines for Improved Software Cost Estimation. Doty Assocates Inc, RADC-TR-77-220, February 1977 (AD-A042 264)

16. James, Thomas G. Software Cost Estimating Methodology. Final Report June 1976--September 1976. AFAL-TR-77-66. Air Force Avionics Laboratory, Wright-Patterson AFB OH. August 1977.

17. James, Thomas G. and Daniel V. Ferens. Application of the RCA Price-S Software Cost Estimation Model to Air Force Avionics Laboratory Programs. Final Report Jan 1978 - July 1979. AFAL-TR-79-1164. Air Force Avionics Laboratory, Wright-Patterson AFB OH. October 1979.

18. Jensen, Randall W. "A Macro-Level Software Development Cost Estimation Methodology," Asilomar Conference on Circuits, Systems and Computers. 320-325. IEEE Press, New York, 1981.

19. Jensen, Randall W. "A Comparison of the Jensen and COCOMO Schedule and Cost Estimation Models," Proceedings of the International Society of Parametric Analysts. 97-106. ISPA Publications, San Francisco, 1984.

20. Killingsworth, Paul, Cost Analyst Space Division. Telephone Interview. Los Angeles AB CA, 12 February 1984.

21. Lorenzetti, Robert C. "Softech Software Cost Estimation Methods," National Aerospace and Electronics Conference. 1041-1048. IEEE Press, New York, 1983.

22. McCallam, Lennis H. "A Comparison of Three Software Costing Techniques," National Aerospace and Electronics Conference. 1021-1026. IEEE Press, New York, 1983.

23. Mullins, James P. "The Commander's Perspective," Skywriter. Wright-Patterson AFB OH. 5-S, 13 January 1984.

24. Naval Avionics Center. Life Cycle Cost Model Estimation Trade Study. Naval Avionics Center, Indianapolis IN. September 1983.

25. Nelson, E. A. Management Handbook for the Estimation of Computer Programming Costs, System Development Corporation, Santa Monica CA. 1967 (AD-648 650).

26. Nie, Norman H., and others. Statistical Package For The Social Sciences. New York: McGraw-Hill, 1975

27. Putnam, Lawrence H. and Ann Fitzsimmons. "Estimating Software Costs," Datamation, a three part series, 189-192 (September 1979), 171-174 (October 1979), 137-139 (November 1979).

28. Quantitative Management, Inc. SLIM User's Manual. QSM, McLean VA, July 1984.

29. RCA/PRICE Systems. PRICE S Reference Manual. RCA, Cherry Hill NJ, undated.

30. Schneider, John A Preliminary Calibration of the RCA Price-S Software Cost Estimation Model. MS Thesis, GSM/SM/77S-15. School of Engineering, Air Force Institute Of Technology. (AU), Wright-Patterson AFB OH, September 1977 (AD-A046 808).

31. Shapiro, O. Software Acquisition Process (SWAP). Report No. MTR-8524, ESD-TR-82-258. Mitre Corporation, Bedford MA. December 1982.

32. Simpson, Robert. Cost Analyst Armament Division. Telephone Interview. Eglin AFB FL, 13 February 1984.

33. Stanley, M. Software Cost Estimating. Report No. DRIC-BR-84024. Royal Signals and Radar Establishment, Malvern, England. 13 May 1982 (AD-A124 258).

34. Steffey, Raymond E. An Analysis of the RCA Price-S Cost Estimation Model as it Relates to Current Air Force Computer Software Acquisition and Management. MS Thesis. GSM/SM/79D-20. School of Engineering, Air Force Institute of Technology. (AU), Wright-Patterson AFB, OH. December 1979 (AD-A083 713).

35. Tausworth, R. L. Deep Space Network Software Cost Estimating Model. Report No. NASA-CR-164277. Jet Propulsion Laboratory, Pasadena CA. April 1981.

36. Thibodeau, Robert. An Evaluation of Software Cost Estimating Models. RADC-TR-81-144. General Research Corporation, Los Angeles CA. June 1981.

37. Wolverton, R.W. "The Cost of Developing Large Scale Software," Transactions on Computers, C-23 (6): 615-636. IEEE Press, New York, June 1974.

VITA

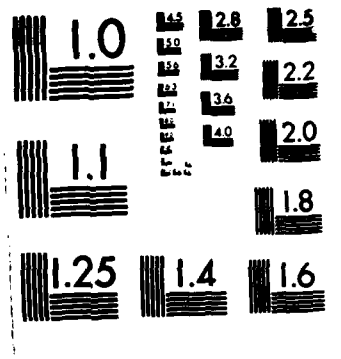
Captain Jeffrey T. Steig was born on 21 September 1954 in Philadelphia, Pennsylvania. He graduated from high school in Honolulu, Hawaii in 1972 and attended the United States Air Force Academy from which he received the degree of Bachelor of Science in International Affairs of the Middle East in June 1976. Upon graduation, he was commisioned and attended pilot training at Reese Air Force Base, Texas. He graduated from pilot training in August 1977 and served as an Instructor Pilot at Reese Air Force Base until August 1980. He then served as a Pilot Training Instructor at Randolph Air Force Base, Texas until entering the School of Systems and Logistics, Air Force Institute of Technology, in May 1983.

Permanent address: 103 Willoughby
Greenville, North Carolina
27834

AD-A147 632 AN APPLICATION OF DISCRIMINANT ANALYSIS TO THE SELECTION OF SOFTWARE COST..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF SYST.. J T STEIG

UNCLASSIFIED SEP 84 AFIT/GSM/LSY/84S-26 F/G 14/1 NL

2/2



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GSM/LSY/84S-26	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Systems and Logistics 6b. OFFICE SYMBOL (If applicable) AFIT/LS	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB OH 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) See Box 19	
12. PERSONAL AUTHOR(S) Jeffrey T. Steig, B.S., Capt, USAF	
13a. TYPE OF REPORT MS Thesis 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 1984 September 15. PAGE COUNT 84
16. SUPPLEMENTARY NOTATION	Approved for public release: IAW AFR 190-17. LYNN E. WOLAVER Dean for Research and Professional Development Air Force Institute of Technology (ATC) Wright-Patterson AFB OH 45433 14 Sept 84

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	
14	01	Cost Effect.	Software Cost Estimating, Cost Estimating Models, Comparison of Software Cost Estimating Models, Software Development Costs, Selecting Software Cost Models
12	01	Math and Stat.	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: AN APPLICATION OF DISCRIMINANT ANALYSIS TO THE SELECTION OF SOFTWARE COST ESTIMATING MODELS

Thesis Chairman: Richard L. Murphy
Assistant Professor

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Richard L. Murphy	22b. TELEPHONE NUMBER (Include Area Code) (513) 255-6280 22c. OFFICE SYMBOL AFIT/LSQ

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

This investigation attempted to improve software cost estimating through the use of discriminant analysis. Currently, no quantitative methods exist to quantitatively select the best software cost estimating model for a particular software type or environment. By identifying the characteristics of the software that each model was best able the estimate, those characteristics could be used as a basis for predicting the best model.

The analysis began by using selected models to concurrently estimate development costs for 25 known projects. The estimates from each model were compared and the most accurate model for each project was identified. The projects were assigned to the group of projects for which each model most accurately estimated development costs. The groups were given the names of the respective models: PRICE S, SLIM, COCOMO, and JENSEN.

After grouping each project, discriminant analysis was used to identify those input variables from all the estimating models that best discriminated between the groups. The identified input variables were then used as determinant variables as a basis to predict which model was most likely to best estimate cost for each project. The unbiased prediction rate was 60%. Despite the high prediction rate, the overall estimating accuracy was not reduced . The criteria statistic, relative root mean squared error, RRMSE, for the estimates selected by discriminant analysis was not significantly better than the RRMSE for the expected error. The results indicated that the use of the pre-analysis determinants to select a model would not reduce estimating error more than a random selection of models.